Funcions predefinides

Utilitats Funcions

Les funcions en qualsevol llenguatge són una sèrie de procediments o instruccions, que realitzen una acció, un avantatge de les funcions és que permeten tenir un codi més ordenat i fàcil de mantenir.

Un altre avantatge de les funcions és que són reutilitzables, pots tenir una funció que validi un formulari i utilitzar-la en tots els teus formularis, pots tenir també una funció que envieu dades al servidor i reutilitzar-la múltiples vegades...

Javascript té més de 4000 funcions predefinides .. Ja n'hem vist algunes: alert, prompt, parseInt, typeOf, Math.round ....

Declaració

Les funcions es componen de quatre parts:

- La paraula clau function que s'utilitza per definir funcions.
- Un nom que ha de ser un identificador vàlid

- Una llista separada per comes de noms de paràmetres entre parèntesis. No totes les funcions requereixen de paràmetres. Si és així, els parèntesis es deixen buits.
- El cos de la funció. Una sèrie d'instruccions entre claus que s'executen quan invoquem la funció.
- Opcionalment, la paraula clau return que s'utilitza per retornar un valor a la sentència que hagi invocat la funció.

> *Nomenclatura: Només poden utilitzar-se lletres, números o el caràcter "_", Ha de ser únic en el codi JavaScript de la pàgina web, No pot començar per un número, ○ No pot tenir paraules reservades.(?)*

Declaració bàsica:

```
function nomFuncio(parametre1,parametre2, parametre3) {
      //[instruccions]
      return valor;
}
```

> *Curiositat: Les funcions,, s'emmagatzemen com a mètodes de l'objecte window..*

Javascript Modern: Expressió de funció

Aquest tipus de funcions s'assigna com si fos una variable.

```
const sumar2 = function() {
    console.log(3 + 3);
}
```

> *Aquesta es la Base de les funcions en el Java Modern.*

Invocacions / crides

Un cop s'ha definit la funció és necessari invocar-la per a que s'executin les instruccions. Per a invocar-la només cal escriure el nom de la funció seguit dels parèntesis. Si la funció disposa de paràmetres, s'han d'especificar al mateix ordre que s'han definit a la funció.

```
nomFuncio(parametre1, parametre2);
```

També podem invocar una funció i passar-li més paràmetres dels inicialment declarats sense que doni error:

```
nomfuncio(parametre1, parametre2, parametre3, parametre4);
```

En canvi, si invoquem una funció sense passar-li els paràmetres esperats, aquests s'estableixen com a undefined.

```
nomfuncio(parametre1);
```

> *Curiositat: A totes les invocacions de les funcions es passen dos paràmetres extres de manera ocultes al programador: 'arguments' i 'this'.*
>
> - *arguments: conté tots els paràmetres que se li passen explícitament a la funció.*
> - *This: Fa referència a l'objecte window (conceptualemt diferent en altres llenguatges de programació)*

This

Per referencia a les variables fora de la funció.

```
var num = 10;

   function canviaNum(num) {
       //canvio el valor del paràmetre num
       num = 20;

       //canvio el valor de num de l'objecte global
       this.num = 30;

       console.log("valor num variable local: " + num);
       console.log("valot this.num variable global: " + this.num);

   }

canviaNum(1)
```

Funcions vs Mètodes

Encara que en conceptualment acaben sent pràcticament el mateix, la manera com s'utilitzen té a veure més que res en el context que són utilitzades.

```
const numero1 = 20;
const numero2 = "20";

console.log( parseInt(numero2) ); // Esto es una función
```

```
console.log( numero1.toString()); // Esto es un método
```

La diferencia és: que mentre la funció pren el valor al parentesi, el mètode afegeix un punt seguit del nom

Comunicació entre funcions

Les teves funcions es comunicaran, en lloc de tenir una gran funció amb 800 línies és recomanable dividir-la en petites parts. Ho podríem pensar en com estructurar el codi en funcionalitats.

```
iniciarApp();

function iniciarApp() {
    console.log('Iniciando App...');
    segundaFuncion();
}

function segundaFuncion() {
    console.log('Desde la segunda función...')
    usuarioAutenticado('Pablo');
}

function usuarioAutenticado(usuario) {
    console.log('Autenticando usuario...');
    console.log(`Usuario autenticado con éxito ${usuario} `)
}
```

Funcions que retornen funcions

Podem tenir funcions que retornen valors per passar-los cap a altres funcions o fer alguna cosa més. Analitzeu i practiqueu amb el següent exercici, que pasa?

```
let total = 0;
function agregarCarrito(precio) {
    return total += precio;
}

function calcularImpuesto(total) {
    return 1.15 * total;
}

total = agregarCarrito(200);
total = agregarCarrito(300);
total = agregarCarrito(400);
```

```
console.log(total);

const totalPagar = calcularImpuesto(total);
console.log(`El total a pagar es de ${totalPagar}`);
```

Tastet arrow function, Java script modern

Els arrow functions són una altra forma de declarar funcions i van ser agregades a les últimes versions, la sintaxi és més curta i al començar a utilitzar-les semblava una mica complexa.

Exemple arrow function Bàsic

```
const aprendiendo = function() {
    console.log('Aprendiendo JavaScript');
}

// arrow function
const aprendiendo = () =>  {
    console.log('Aprendiendo JavaScript');
}
```

Exemple arrow function en una linea

```
// si esribim tot en una linea no requereix claus
const aprendiendo = () => console.log('Aprendiendo JavaScript');

// retorna un valor
const aprendiendo = () => 'Aprendiendo JavaScript';

console.log(aprendiendo());
```

Exemple arrow function amb paràmetres

```
// Parametros
const aprendiendo = (tecnologia) => console.log(`aprendiendo ${tecnologia}`);
aprendiendo('JavaScript');

// si es un solo parmetro no ocupamos el parentesis
const aprendiendo = tecnologia => console.log(`aprendiendo ${tecnologia}`);
aprendiendo('JavaScript');

// multiples parametros si requieren parentesis
const aprendiendo = (tecn1, tecn2) => console.log(`Aprendiendo ${tecn1}
${tecn2}`);
aprendiendo('JS', 'ES');
```

Exercicis

1. Crea una funció que es digui Saludar, que tingui tres argument: nom, cognom, sexe. I que escrigui per pantalla:
   Benvinguda [nom] [Cognoms] si és dona.
   Benvingut [nom] [Cognoms] si és home.

2. Crea una funció que es calculi l'edat d'un gos. Quin parametre podries fer servir?
3. Crea una funció on se li pasa 2 números. Que retorni el menor.
4. Programa en JavaScript una funció on se li passi dos arguments: string1 -frase- i string2-caràcter-, i retorna quants caràcters hi ha en total = string1 + string2.
5. Analitza e investiga les diferencies entre les següents funcions. Aquesta és una pregunta de entrevista com a JS Developer. (Hoisting)

```
sumar();
function sumar() {
    console.log(2 + 2);
}

sumar2();
const sumar2 = function() {
    console.log(3 + 3);
}
```

6. Donada una adreça de correu, implementa una funció que verifiqui si el email té carregat el caràcter @.

7. Carregar un String per teclat i implementar les funcions següents:
   a. Imprimir la primera meitat dels caràcters de la cadena.
   b. Imprimir el darrer caràcter.
   c. Imprimir-ho en forma inversa.
   d. Imprimir cada caràcter del String separat amb un guió.
   e. Imprimir la quantitat de vocals emmagatzemades.